B5

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/778,076	02/07/2001	Yutaka Haga	1046.1236/JDH	4573

21171 7590 01/30/2004

STAAS & HALSEY LLP
SUITE 700
1201 NEW YORK AVENUE, N.W.
WASHINGTON, DC 20005

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2122	4

DATE MAILED: 01/30/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/778,076	HAGA, YUTAKA
	Examiner	Art Unit
	Michael J. Yigdall	2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 February 2001 and 17 April 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-39* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-39* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *07 February 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

13)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 3 .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. Claims 1-39 are pending and have been examined. The priority date considered for the application is 26 September 2000.

Specification

2. The disclosure is objected to because of the following informalities: The application appears to be a literal translation into English from a foreign document and contains grammatical and/or typographical errors. See, for example, in the last paragraph on page 2, "to collecting and analyze profiles," which should be replaced with --to collect and analyze profiles--. Also see, for example, in the last paragraph on page 8, "it is possible to writes," which should be replaced with --it is possible to write--. Appropriate correction is required. Applicant is respectfully asked for cooperation in finding all such grammatical informalities and ensuring that the specification and claims are clear and concise.

Claim Objections

3. Claims 12, 13, 15, 16, 18, 23, 24, 26, 27, 29, 34, 35, 37 and 38 are objected to because of the following informalities: There are several instances of "brunch instruction," "brunch source address" and "brunch destination address" recited in the claims. The word "brunch" in each case should be replaced with --branch--. Appropriate correction is required. The claims have been interpreted assuming this correction to be made.

4. Claim 18 is objected to because of the following informalities: The claim language is generally indefinite and contains grammatical and idiomatic errors. See, for example, "a computer readable medium stores a program in order that a computer executes a process for

collecting a profile of a subroutine included another program as an analizing target," which appears to be a literal translation into English from a foreign document. Appropriate correction is required. The claim has been interpreted accordingly.

5. Claim 19 is objected to as being in improper form because, as written in the application, it is dependent upon itself. Appropriate correction is required. The claim has been interpreted assuming it to depend from claim 18.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

7. Claims 1, 18 and 29 are rejected under 35 U.S.C. 102(a) as being anticipated by Japanese Patent Laid-Open Pub. No. 2000-20354 to Akita et al. (English translation).

With respect to claim 1, Akita et al. discloses an apparatus for collecting a profile of a subroutine in a program, the apparatus collecting the profile of said subroutine by using an interrupt generated when a branch instruction is executed during execution of said program (see page 1, Solving Means, which shows generating an interrupt for each branch instruction encountered during program execution and preparing a file having profile information).

With respect to claim 18, Akita et al. discloses a computer readable medium stores a program in order that a computer executes a process for collecting a profile of a subroutine

included another program as an analyzing target (see page 1, Solving Means, which shows collecting trace records and preparing a profile based on the execution of a subroutine), said program comprising the steps of:

(a) identifying, when an interrupt is generated by an execution of a branch instruction during the execution of said other program, whether or not said executed branch instruction is an instruction relating to the execution of said subroutine (see page 3, paragraph 15, which shows generating an interrupt upon the execution of a branch instruction; see also page 4, paragraph 22, which shows identifying whether the branch instruction relates to a function call);

(b) obtaining, when said branch instruction is the instruction relating to the execution of said subroutine, said profile of said subroutine (see page 3, paragraph 15, which shows obtaining a profile of a subroutine); and

(c) storing said profile in a storage unit (see page 3, paragraph 15, which shows storing the profile in a trace record file, i.e. a storage unit).

With respect to claim 29, Akita et al. discloses a method for collecting a profile of a subroutine in a program (see page 1, Solving Means, which shows collecting trace records and preparing a profile based on the execution of a subroutine), comprising the steps of:

(a) identifying, when an interrupt is generated by an execution of a branch instruction during the execution of the program, whether or not the executed branch instruction is an instruction relating to the execution of the subroutine (see page 3, paragraph 15, which shows generating an interrupt upon the execution of a branch instruction; see also page 4, paragraph 22, which shows identifying whether the branch instruction relates to a function call);

(b) obtaining, when the branch instruction is the instruction relating to the execution of the subroutine, the profile of the subroutine (see page 3, paragraph 15, which shows obtaining a profile of a subroutine); and

(c) storing the profile in a storage unit (see page 3, paragraph 15, which shows storing the profile in a trace record file, i.e. a storage unit).

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1-39 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Pat. No. 6,002,872 to Alexander, III et al. (hereinafter Alexander) in view of U.S. Pat. No. 6,253,338 to Smolders.

With respect to claim 1, Alexander discloses an apparatus for collecting a profile of a subroutine in a program, the apparatus collecting the profile of said subroutine by using an interrupt (see the title and abstract).

Alexander does not disclose the limitation wherein the interrupt is generated when a branch instruction is executed during execution of said program.

Alexander does show using timer interrupts (see column 4, lines 20-23) and also shows that other interrupts may be used instead (see column 11, lines 22-25).

Smolders discloses the limitation above in terms of generating a trace interrupt after every branch instruction (see column 3, lines 58-61) in a performance monitor (see the title and abstract), so as to enable tracing without introducing any overhead and without modifying the code (see column 1, lines 64-67).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use a branch interrupt as taught by Smolders in place of the timer interrupt disclosed by Alexander for the purpose of enabling tracing and profiling without introducing overhead or modifying the program code.

With respect to claim 2, Alexander further discloses an apparatus that collects said profile for each executor of a subroutine (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles; note the parent node pointers, which identify each executor of a subroutine).

With respect to claim 3, Alexander further discloses an apparatus that individually collects profiles of a plurality of subroutines executed by a specific executor (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles; note that a path in the tree represents the subroutines executed by a specific executor).

With respect to claim 4, Alexander further discloses an apparatus that individually collects a first profile of a subroutine called by a main routine, and a second profile of this subroutine called by another routine (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles; see also FIG. 10 and column 8, lines 24-40, which shows

individual profiles of subroutines organized based on the calling routine; see also FIG. 8, which shows, for example, a subroutine Y called by both the main routine and by a second routine X).

With respect to claim 5, Alexander further discloses an apparatus that stores said second profile and calling relationship information relating to said second profile, said calling relationship information indicating a relationship between said other subroutine and said subroutine (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including calling relationship information).

With respect to claim 6, Alexander further discloses the limitation wherein said profile of said subroutine comprises at least one from a cumulative value of execution time, times of calling, time of final calling, and an overhead (see column 6, lines 37-53, which shows that the profile comprises both a base time and a cumulative execution time).

With respect to claim 7, Alexander further discloses:

(a) a storage unit storing a profile of a subroutine (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles; note that each node is considered a storage unit);

(b) an analyzing section judging whether or not an executed branch instruction is an instruction relating to the execution of said subroutine when said interrupt is generated (see column 5, lines 20-32, which shows analyzing the stack frames in response to an interrupt to identify subroutines); and

(c) a collecting section, when said analyzing section identifies that said executed branch instruction is the instruction relating to the execution of said subroutine, for obtaining a profile of

said subroutine and stores said obtained profile in said storage unit (see column 5, lines 33-40, which shows obtaining samples or profiles and storing them).

With respect to claim 8, Alexander further discloses the limitation wherein a plurality of storage units respectively corresponding to a plurality of executors of said subroutine are prepared (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a plurality of nodes or storage units); and

said collecting section specifies said executor of said subroutine and stores said profile of said subroutine corresponding to said specified executor in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles; note the parent node pointers, which specify the executors of subroutines).

With respect to claim 9, Alexander further discloses the limitation wherein said collecting section individually stores profiles of a plurality of subroutines corresponding to a specified executor in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles; note that a path in the tree represents the subroutines executed by a specific executor).

With respect to claim 10, Alexander further discloses the limitation wherein said collecting section individually stores a first profile of a subroutine called by a main routine, and a second profile of the subroutine called by another subroutine, in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles; see also FIG. 10 and column 8, lines 24-40, which shows individual profiles of subroutines organized based on

the calling routine; see also FIG. 8, which shows, for example, a subroutine Y called by both the main routine and by a second routine X).

With respect to claim 11, Alexander further discloses the limitation wherein said collecting section stores said second profile and calling relationship information relating to said second profile, said calling relationship information indicating a relationship between said other subroutine and said called subroutine, in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including calling relationship information).

With respect to claim 12, Alexander further discloses:

(a) a storage unit storing a profile (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles; note that each node is considered a storage unit);

(b) an analyzing section, when said interrupt generates, obtaining a branch source address and a branch destination address from a source of said interrupt (see column 5, lines 20-32, which shows analyzing the stack frames in response to an interrupt to identify subroutines; see also column 5, lines 41-62, which shows obtaining a call or branch source address and a return or branch destination address).

Alexander does not disclose identifying a type of said branch instruction by obtaining an instruction code from said branch source address and decoding said instruction code.

Smolders discloses the limitation above in terms of identifying branch instructions (see column 4, lines 1-6; note that instruction codes are inherently decoded in order to distinguish between branch instructions and non-branch instructions) in a performance monitor (see the title

and abstract), so as to enable tracing without introducing any overhead and without modifying the code (see column 1, lines 64-67).

Alexander further discloses:

(c) a collecting section obtaining said branch source address, said branch destination address, and a identified result from said analyzing section when the identified instruction is a calling instruction or a return instruction of said subroutine (see column 5, lines 41-62, which shows obtaining a call or branch source address and a return or branch destination address for a subroutine); when said identified result is said calling instruction, storing said branch destination address as a subroutine address corresponding to said calling instruction and a calling time of said subroutine corresponding to said calling instruction in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a subroutine address and a base time); and when said identified result is said return instruction, obtaining a return time of said subroutine corresponding to said return instruction, calculating a execution time of said subroutine based on said obtained return time and said calling time, and storing a cumulative value of said execution time as said profile in correspondence with said branch destination address in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including an execution time and a cumulative time, inherently calculated based on the calling time and the return time; see also FIG. 4A, which shows timestamps for entering and returning from subroutines).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use a branch interrupt as taught by Smolders in place of the timer interrupt

disclosed by Alexander for the purpose of enabling tracing and profiling without introducing overhead or modifying the program code.

With respect to claim 13, Alexander further discloses the limitation wherein said collecting section stores times of calling of said subroutine corresponding to said branch destination address as said profile in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a subroutine address and a base time; see also FIG. 4A, which shows timestamps for entering, i.e. times of calling, and returning from subroutines).

With respect to claim 14, Alexander further discloses the limitation wherein said collecting section obtains an overhead of said subroutine as said profile and stores said overhead in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a time consumed by a thread executing a subroutine, which is considered a measure of overhead).

With respect to claim 15, Alexander further discloses the limitation wherein said collecting section, when said identified result is said calling instruction, stores an identifier of an executor of said subroutine corresponding to said calling instruction and said branch destination address in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a subroutine address; note the parent node pointers, which identify the executors of subroutines).

With respect to claim 16, Alexander further discloses the limitation wherein said collecting section, when said identified result is said calling instruction and said branch source address and said branch destination address are addresses of said subroutines, stores said branch source address and branch destination address as calling relationship information indicating a callings source subroutine and a calling destination subroutine is said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a subroutine address and calling relationship information), and stores at least one of the cumulative execution time and the times of calling in said calling destination subroutine in the call source subroutine, as said profile corresponding to said calling relationship information, in said storage unit (see column 6, lines 37-53, which shows that the profile comprises both a base time and a cumulative execution time).

With respect to claim 17, Alexander does not disclose a setting section setting an execution environment of a source of said interrupt so as to generate said interrupt when said branch instruction is executed during the execution of said program.

Alexander does show using timer interrupts (see column 4, lines 20-23) and also shows that other interrupts may be used instead (see column 11, lines 22-25).

Smolders discloses the limitation above in terms of setting the execution environment to generate a trace interrupt after every branch instruction (see column 3, lines 58-61) in a performance monitor (see the title and abstract), so as to enable tracing without introducing any overhead and without modifying the code (see column 1, lines 64-67).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use a branch interrupt as taught by Smolders in place of the timer interrupt

disclosed by Alexander for the purpose of enabling tracing and profiling without introducing overhead or modifying the program code.

With respect to claims 18 and 29, Alexander discloses a computer readable medium storing a program and a method for collecting a profile of a subroutine included in another program (see the title and abstract; see also column 11, lines 6-17).

Alexander does not disclose the step of:

(a) identifying, when an interrupt is generated by an execution of a branch instruction during the execution of said other program, whether or not said executed branch instruction is an instruction relating to the execution of said subroutine.

Alexander does show using timer interrupts (see column 4, lines 20-23) and also shows that other interrupts may be used instead (see column 11, lines 22-25).

Smolders discloses the limitation above in terms of generating a trace interrupt after every branch instruction (see column 3, lines 58-61) and identifying branch instructions (see column 4, lines 1-6; note that instruction codes are inherently decoded in order to distinguish between branch instructions and non-branch instructions) in a performance monitor (see the title and abstract), so as to enable tracing without introducing any overhead and without modifying the code (see column 1, lines 64-67).

Alexander further discloses:

(b) obtaining, when said branch instruction is the instruction relating to the execution of said subroutine, said profile of said subroutine (see column 5, lines 33-40, which shows obtaining samples or profiles and storing them); and

(c) storing said profile in a storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use a branch interrupt as taught by Smolders in place of the timer interrupt disclosed by Alexander for the purpose of enabling tracing and profiling without introducing overhead or modifying the program code.

With respect to claims 19 and 30, Alexander further discloses the steps of:

(a) specifying an executor of said subroutine (see column 5, lines 41-62, which shows identifying or specifying the process and thread of an executor); and

(b) storing said profile in a storage unit corresponding to said specified executor within a plurality of storage unit provided with each executor (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a plurality of nodes or storage units).

With respect to claims 20 and 31, Alexander further discloses the step of storing, when a specified executor executes a plurality of subroutines, profiles of said plurality of subroutines in said storage unit corresponding to said specified executor (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles; note that a path in the tree represents the subroutines executed by a specific executor).

With respect to claims 21 and 32, Alexander further discloses the step of storing a first profile of said subroutine called by a main routine and a second profile of said subroutine called by another subroutine in said corresponding storage unit, regarding to each of said subroutines

(see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles; see also FIG. 10 and column 8, lines 24-40, which shows individual profiles of subroutines organized based on the calling routine; see also FIG. 8, which shows, for example, a subroutine Y called by both the main routine and by a second routine X).

With respect to claims 22 and 33, Alexander further discloses the step of storing said second profile and calling relationship information relating to said second profile, said calling relationship information indicating a relationship between said other subroutine and said subroutine (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including calling relationship information).

With respect to claims 23 and 34, Alexander further discloses the steps of:

(a) obtaining a branch source address and a branch destination address from a source of said interrupt when said interrupt generates (see column 5, lines 20-32, which shows analyzing the stack frames in response to an interrupt to identify subroutines; see also column 5, lines 41-62, which shows obtaining a call or branch source address and a return or branch destination address);

Alexander does not disclose identifying a type of said branch instruction by obtaining an instruction code from said branch source address and decoding said instruction code.

Smolders discloses the limitation above in terms of identifying branch instructions (see column 4, lines 1-6; note that instruction codes are inherently decoded in order to distinguish between branch instructions and non-branch instructions) in a performance monitor (see the title

and abstract), so as to enable tracing without introducing any overhead and without modifying the code (see column 1, lines 64-67).

Alexander further discloses:

(c) storing said branch destination address as a subroutine address corresponding to said calling instruction and a calling time of said subroutine corresponding to said calling instruction in said storage unit when said identified result is said calling instruction (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a subroutine address and a base time); and

(d) when said identified result is said return instruction, obtaining a return time of said subroutine corresponding to said return instruction, calculating a execution time of said subroutine based on said obtained return time and said calling time, and storing a cumulative value of said execution time as said profile in correspondence with said branch destination address in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including an execution time and a cumulative time, inherently calculated based on the calling time and the return time; see also FIG. 4A, which shows timestamps for entering and returning from subroutines).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use a branch interrupt as taught by Smolders in place of the timer interrupt disclosed by Alexander for the purpose of enabling tracing and profiling without introducing overhead or modifying the program code.

With respect to claims 24 and 35, Alexander further discloses the step of storing times of calling of said subroutine corresponding to said branch destination address as said profile in said

storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a subroutine address and a base time; see also FIG. 4A, which shows timestamps for entering, i.e. times of calling, and returning from subroutines).

With respect to claims 25 and 36, Alexander further discloses the step of storing an overhead of said subroutine as said profile in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a time consumed by a thread executing a subroutine, which is considered a measure of overhead).

With respect to claims 26 and 37, Alexander further discloses the step of storing, when said identified result is said calling instruction, and identifier of an executor of said subroutine corresponding to said calling instruction and said branch destination address in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a subroutine address; note the parent node pointers, which identify the executors of subroutines).

With respect to claims 27 and 38, Alexander further discloses the steps of, when said identified result is said calling instruction and said branch source address and said branch destination address are addresses of said subroutines, storing said branch source address and branch destination address as calling relationship information indicating a calling source subroutine, and a calling destination subroutine in said storage unit (see FIG. 5 and column 6, lines 37-53, which shows the data structure used to store the profiles, including a subroutine address and calling relationship information); and storing at least one of the cumulative execution time and the times of calling in said calling destination subroutine in the call source

subroutine, as said profile corresponding to said calling relationship information, in said storage unit (see column 6, lines 37-53, which shows that the profile comprises both a base time and a cumulative execution time).

With respect to claims 28 and 39, Alexander does not disclose the step of setting an execution environment of a source of said interrupt so as to generate said interrupt when said branch instruction is executed during the execution of said program.

Alexander does show using timer interrupts (see column 4, lines 20-23) and also shows that other interrupts may be used instead (see column 11, lines 22-25).

Smolders discloses the limitation above in terms of setting the execution environment to generate a trace interrupt after every branch instruction (see column 3, lines 58-61) in a performance monitor (see the title and abstract), so as to enable tracing without introducing any overhead and without modifying the code (see column 1, lines 64-67).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use a branch interrupt as taught by Smolders in place of the timer interrupt disclosed by Alexander for the purpose of enabling tracing and profiling without introducing overhead or modifying the program code.

Conclusion

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Japanese Patent Laid-Open Pub. No. 4-102942 to Murakami (English translation) discloses profiling based on branch instruction interrupts.

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (703) 305-0352. The examiner can normally be reached on Monday through Friday from 8:00am to 4:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (703) 305-4552. The fax phone number for the organization where this application or proceeding is assigned is (703) 746-7239.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

MY
Michael J. Yigdall
Examiner
Art Unit 2122

mjy
January 15, 2004

TUAN DAM
SUPERVISORY PATENT EXAMINER